# Ensuring the Correctness of Business Workflows at the Syntactic Level: An Ontological Approach

Thi-Hoa-Hue Nguyen[1,2] * and Nhan Le-Thanh[2]

[1] Information Technology Faculty
Vietnam-Korea Friendship Information Technology College
Da Nang, Vietnam
[2] WIMMICS - The I3S Laboratory - CNRS - INRIA
Nice Sophia Antipolis University
Sophia Antipolis, France
huenth@gmail.com,Nhan.LE-THANH@unice.fr

**Abstract.** High quality business workflow definitions play an important role in the organization. An incorrectly defined workflow may lead to unexpected results. Therefore, each business workflow definition should be carefully analyzed before it is put into use. In this paper, we propose an ontological approach which is suitable for ensuring the syntactic correctness of business workflows. In details, to represent CPNs with OWL DL, we first introduce the CPN Ontology. Then, we define axioms, which are added to the CPN Ontology to provide automated support for establishing the correctness of business workflows. Finally, by relying on the CORESE semantic engine, SPARQL queries are implemented to detect shortcomings in concrete workflows. To the best of our knowledge, this is a novel approach for the representation and verification of business workflows based on ontologies.

**Keywords:** Business Workflow, Correctness, OWL DL, SPARQL, Verification

## 1 Introduction

The current tendency in e-business has resulted in more complex business processes. However, the specification of a real-world business process is generally manual and is thus vulnerable to human error. An incorrectly designed workflow may lead to failed workflow processes, execution errors or not meet the requirements of customers, etc. In fact, existing techniques applied to check the correctness of workflows are particularly used in commercial business workflow systems. Most of them assume that a workflow is correct if it complies with "the constraints on data and control flow during execution" [1]. Whether the workflow is in conformity with the design requirements is neither specified nor proved. There is thus a great need for developing a thorough and rigorous method

* This work was done as part of a collaboration between Nice Sophia Antipolis University and Da Nang University

that automatically supports workflow designers to ensure workflows being well-formed.

In this study, we extend our previous work [2] in designing well-formed CPNs-based business workflow templates (CBWTs) and checking their correctness. The approach is based on Knowledge Engineering, Coloured Petri Nets (CPNs) and Semantic Web technologies which provide semantically rich business process definitions and automated support for CBWTs verification. Our contributions are:

- Presenting a classification of syntactic constraints in modelling business processes and creating their related axioms using Description Logic (DL) in order to support workflow designers;
- Showing the SPARQL [3] query language is able to verify workflow templates.

The rest of this paper is structured as follows: In Section 2, we briefly introduce our CPN Ontology as a representation Coloured Petri Nets (CPNs) with OWL DL. We then present syntactic constraints and create their related axioms added to the CPN Ontology to support designers in establishing well-formed workflow templates in Section 3. In Section 4, we introduce the SPARQL query language used to verify CBWTs at the syntactic level. In Section 5 we give related work. Finally, Section 6 concludes the paper with an outlook on the future research.

## 2 Modelling Business Processes with Coloured Petri Nets - The CPN Ontology

On one hand, Coloured Petri Nets (CPNs) [4] have been developed into a full-fledged language for the design, specification, simulation, validation and implementation of large-scale software systems. CPN is a well-proven language which is suitable for modelling workflows or work processes. Therefore, CPNs are chosen as the workflow language in our work to transform a business process into a control flow-based business workflow template. However, it is difficult to interoperate, share and reuse business processes modelled with CPNs, i.e., business workflows, because of the lack of semantic representation of CPN components [2].

On the other hand, an ontology with its components, which provides machine-readable definitions of concepts, can represent semantically rich workflow definitions. Once workflow definitions are stored as semantically enriched workflow templates, developers can easily build their appropriate software systems from these templates. Therefore, in this section, we shortly introduce the CPN Ontology, which is first proposed in [2] as a representation of CPNs with OWL Description Logic (OWL DL). The main purpose is to facilitate business processes modelled with CPNs to be easily shared and reused.

In order to develop the CPN Ontology, we translate each element of CPNs into a corresponding OWL concept. The core concepts of the CPN ontology is

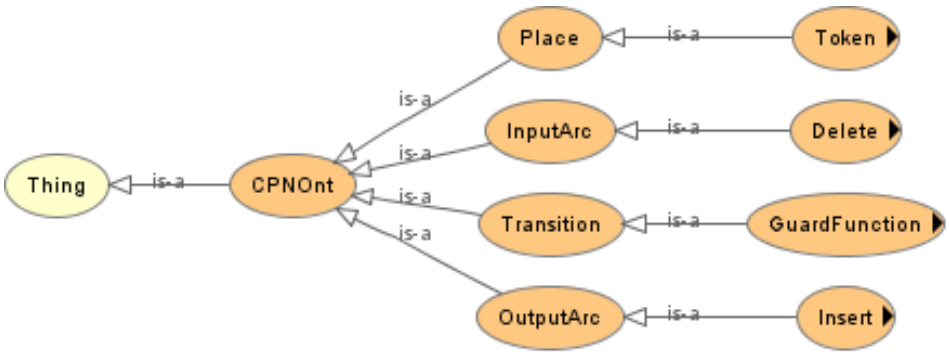


Fig. 1: CPN Ontology (excerpt)

depicted in Figure 1. The ontology is described based on DL syntax and the axioms supported by OWL.

In the CPN Ontology, we define the concept **CPNOnt** for all possible business processes modelled with CPNs. We define the concept **Place** and the concept **Transition** to represent all places and transitions of a process model, respectively. In order to represent all directed arcs from places to transitions and all directed arcs from transitions to places, we define the concept **InputArc** and the concept **OutputArc**, respectively. In our case, one place contains no more than one token at one time, therefore, the concept **Token** is defined for all tokens inside places. To express all transition expressions, the concept **GuardFunction** is defined. Transitions consist of control and activity nodes. We define the concept **CtrlNode** for occurrence condition in the former nodes and the concept **ActNode** for occurrence activity in the latter nodes. The concepts **Delete** and **Insert** are defined for all expressions in input arcs and output arcs, respectively. In order to express all attributes of individuals, we define the concept **Attribute**. And for all subsets of $I_1 \times I_2 \times \ldots \times I_n$ where $I_i$ is a set of individuals, the concept **Value** is defined.

Properties between the concepts in the CPN Ontology are also indicated. For example, a class **Transition** has two properties **connectsPlace** and **hasGuardFunction**. Consequently, the concept **Transition** can be glossed as "The class **Transition** is defined as the intersection of: (i) any class having at least one property **connectsPlace** whose value is equal to the class **Place** and; (ii) any class having one property **hasGuardFunction** whose value is restricted to the class **GuardFunction**" [2].

## 3 Taxonomy of Constraints in Modelling Business Processes

To provide automated support for workflow designers in establishing the correctness of ontology-based workflow representations, in this Section we introduce a

set of syntactic constraints. The constraints are categorized into two groups. Axioms related to the constraints are also defined using a DL as $\mathcal{SHOIN}(\mathcal{D})$ to complete the CPN Ontology.

As mentioned earlier, we aim at representing the correct CBWTs in a knowledge base. Therefore, at first, we define the soundness property that is used as the criterion to check the correctness of workflow processes at the syntactic level.

**Definition 1 (Sound).** *A CPN-based process model, $PM$, is sound iff:*

*(i) $PM$ is connected and well-formed;*
*(ii) For every state $M_j$ reachable from state Start $M_0$, there also exists another firing sequence starting from state $M_j$ to state End $M_e$;*
*(iii) State End $M_e$ is the only state which is reachable from state Start $M_0$ with one token in place e;*
*(iv) There is no deadlock, no infinite cycle and no missing synchronization in $PM$.*

### 3.1 Syntactic Constraints related to the Definition of Process Model

– **Constraints related to places.**
**Constraint 1.** For every place $p \in P$, $p$ connects and/or is connected with transitions via arcs.
We create the axiom corresponding to Constraint 1 as follows:
$hasPlace^{-}.CPNOnt \sqcap \neg(\exists connectsTrans.hasTrans^{-}.CPNOnt \sqcup \exists connectsPlace^{-}.hasTrans^{-}.CPNOnt) \sqsubseteq \bot$
**Constraint 2.** There is one and only one start point in a process model.
We create the axiom corresponding to Constraint 2 as follows:
$CPNOnt \sqcap \neg(= 1\ hasPlace.(connectsTrans.hasGuardFunction.hasActivity.ActNoce \sqcap \neg(\exists\ connectsPlace^{-}.hasTrans^{-}.CPNOnt))) \sqsubseteq \bot$
**Constraint 3.** There is one and only one end point in a process model.
We create the axiom corresponding to Constraint 3 as follows:
$CPNOnt \sqcap \neg(= 1\ hasPlace.(connectsPlace^{-}.hasGuardFunction.hasActivity.ActNode \sqcap \neg(\exists\ connectsTrans.hasTrans^{-}.CPNOnt))) \sqsubseteq \bot$
**Constraint 4.** A place has no more than one leaving arc. If a place is connected to a transition, there exists only one directed arc from the place to the transition.
We create the axiom corresponding to Constraint 4 as follows:
$Place \sqcap \neg(\leq 1\ hasPlace^{-}.InputArc) \sqsubseteq \bot$
**Constraint 5.** A place has no more than one entering arc. If a transition is connected to a place, there exists only one directed arc from the transition to the place.
We create the axioms corresponding to Constraint 5 as follows:
$Place \sqcap \neg(\leq 1\ connectsPlace^{-}.(= 1 hasTrans^{-}.OutputArc)) \sqsubseteq \bot$
**Constraint 6.** There are no pairs of activity nodes connected via a place.
We create the axiom corresponding to Constraint 6 as follows:
$Place \sqcap \exists connectsTrans.hasGuardFunction.hasActivity.ActNode \sqcap$

$\exists connectsPlace^{-}.hasGuardFunction.hasActivity.ActNode \sqsubseteq \bot$

**Constraint 7.** There are no pairs of control nodes connected via a place.

We create the axiom corresponding to Constraint 7 as follows:

$Place \sqcap \exists connectsTrans.hasGuardFunction.hasControl.CtrlNode \sqcap \exists connectsPlace^{-}.hasGuardFunction.hasControl.CtrlNode \sqsubseteq \bot$

- **Constraints related to transitions.**

  **Constraint 8.** A transition is on the path from the start point to the end point of a process model.

  - If a transition has no input place, it will never be enabled;
  - If a transition has no output place, it will not lead to the end.

  Consequently, each transition in a workflow must have at least one entering arc and at least one leaving arc.

  We create the axiom corresponding to Constraint 8 as follows:

  $Transition \sqsubseteq \geq 1\ connectsPlace.Place \sqcap \geq 1\ connectsTrans^{-}.Place$

  **Constraint 9.** An activity node has only one entering arc and one leaving arc.

  We create the axiom corresponding to the Constraint 9 as follows:

  $hasGuardFunction.hasActivity.ActNode \sqsubseteq = 1\ connectsPlace.Place \sqcap = 1\ connectsTrans^{-}.Place$

  **Constraint 10.** A control node does not have both multi-leaving arcs and multi-entering arcs.

  We create the axiom corresponding to the Constraint 10 as follows:

  $\geq 2\ connectsPlace.Place \sqcap \geq 2\ connectsTrans^{-}.Place \sqcap hasGuardFunction.hasControl.CtrlNode \sqsubseteq \bot$

- **Constraints related to directed arcs.**

  **Constraint 11.** Directed arcs connect places to transitions or vice versa.

  We create the axioms corresponding to the Constraint 11 as follows:

  $hasPlace^{-}.InputArc \equiv connectsTrans.hasTrans^{-}.CPNOnt$

  $hasTrans^{-}.OutputArc \equiv connectsPlace.hasPlace^{-}.CPNOnt$

### 3.2 Syntactic Constraints Related to Uses of Control Nodes

A poorly designed workflow due to improper uses of control nodes can result in deadlock, infinite cycle or missing synchronization. However, these errors can be detected when designing a workflow template and therefore, we can get rid of them. To do that, we next introduce Constraint 12 and the symptoms related to deadlock, infinite cycle or missing synchronization.

**Constraint 12.** There is no deadlock, no infinite cycle and no missing synchronization.

- **Deadlock**: A deadlock is a situation in which a process instance falls into a stalemate such that no more activity can be enabled to execute. Figure 2 shows three simple deadlock simulations.
- **Infinite cycle**: An infinite cycle is derived from structural errors where some activities are repeatedly executed indefinitely. A simple infinite simulation is depicted in Figure 3(a).

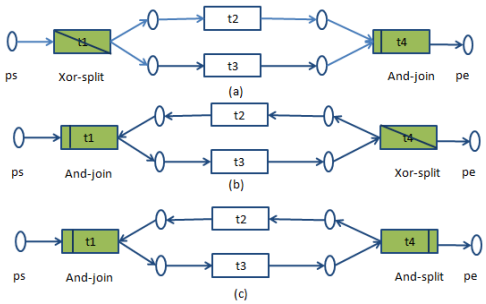


Fig. 2: Deadlock simulations

– **Missing synchronization**: Missing synchronization is a situation in which the mismatch between the building blocks leads to neither deadlock nor infinite cycle, but results in unplanned executions. Figure 3(b) shows a simple simulation of missing synchronization.

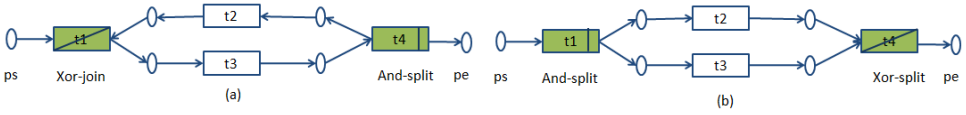


Fig. 3: Infinite cycle simulation

We next create the axioms related to the control nodes (one of two types of transitions), including $And-split$, $And-join$, $Xor-split$ and $Xor-join$, used to detect deadlock, infinite cycle or missing synchronization.

– **And-split** is connected to at least two output places. Every output place contains one token. We create the axiom corresponding to *And-split* as follows:
$AndSplit \sqsubseteq Transition \sqcap connectsPlace.hasMarking.Token \sqcap connectsTrans^{-}.hasMarking.Token \sqcap hasGuardFunction.hasControl.CtrlNode \sqcap =1\ connectsTrans^{-}.Place \sqcap \geq 2\ connectsPlace.Place$
– **And-join**: There are at least two input places connected to *And-join*. In order to activate *And-join*, every input place has to contain one token. We create the axiom corresponding to *And-join* as follows:
$AndJoin \sqsubseteq Transition \sqcap connectsPlace.hasMarking.Place \sqcap connectsTrans^{-}.hasMarking.Token \sqcap hasGuardFunction.hasControl.CtrlNode \sqcap \geq 2 connectsTrans^{-}.Place \sqcap =1\ connectsPlace.Place$
– **Xor-split** is connected to at least two output places. Unlike *And-split*, at any time, one and only one output place of *Xor-join* can contain a token. We create the axiom corresponding to *Xor-split* as follows:

$XorSplit \sqsubseteq Transition \sqcap \neg AndSplit \sqcap hasGuardFunction.hasControl.CtrlNode \sqcap = 1\ connectsTrans^{-}.Place \sqcup \geq 2\ connectsPlace.Place \sqcup connectsTrans^{-}.hasMarking.Token$

- **Xor-join**: There are at least two input places connected to *Xor-join*. Unlike *And-join*, *Xor-join* is activated if one and only one input place contains a token. We create the axiom corresponding to *Xor-join* as follows:
$XorJoin \sqsubseteq Transition \sqcap \neg AndJoin \sqcap connectsPlace.hasMarking.Token \sqcap \geq 2\ connectsTrans^{-}.Place. \sqcap hasGuardFunction.hasControl.CtrlNode \sqcap = 1\ connectsPlace.Place$

### 3.3 A Wrong Workflow Example

An example of a wrongly designed business process modelled with CPNs is illustrated in Figure 4. The air ticket agent first requires a customer to provide some information related to the flights that he or she wants to book, including name(s), depart, destination, date and class. It then looks for the requested ticket(s) on its partner websites. For simplicity, we assume that two websites are utilized. The obtained results, which may consist of no results, some results or time out, are then evaluated in order to make a decision.

As shown in Figure 4, the example model contains syntactic errors. There are three end points, i.e., *Time out*, *End* 2 and *End* 3. Besides, the combination of a Xor-split (the transition $t2$ - *Prepare to look for a flight*) and an And-join (the transition $t5$ - *Collect results*) causes a deadlock. Assuming that the place *Request verified* contains a token that makes the transition $t2$ to be enabled. If the transition Xor-split $t2$ fires, it consumes the token from its input place *Request verified* and then produces one token for only one of its output places. Consequently, either $t3$ or $t4$ may be activated. Since only one of the two transitions $t3$ and $t3$ can fire, not all input places of the transition And-join $t5$ can get its token. As a result, a deadlock occurs because the transition $t6$ will never be enabled to fire.

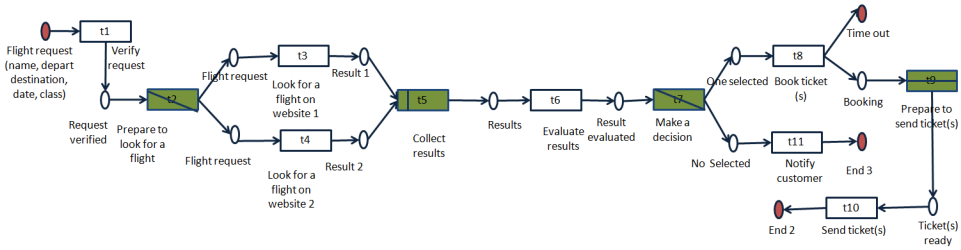


Fig. 4: A wrongly designed workflow model for the airline booking process

We have introduced the CPN ontology represented in OWL DL and axioms which are defined to support designers in verifying CPNs-based process models. It is necessary to note that, to develop or modify CBWTs (i.e., CPN models),

manipulation operations [2], such as inserting new elements, deleting existing elements, etc., on business process models are required. Therefore, at design time, workflow templates stored in RDF format need to be verified before they are put into use. In the next Section, we present the SPARQL query language used to detect shortcomings in workflow templates represented in RDF syntax.

## 4 Using SPARQL to Verify Workflow

The CORESE [5], a semantic search engine, developed for answering SPAQRL queries asked against an RDF knowledge base, is used in our work. We choose the SPAQRL query language because: (i) It is an RDF query language; (ii) It is a W3C Recommendation and is widely accepted in the Semantic Web and also AT community; (iii) Its syntax is quite simple which allows for a query to include triple patterns, conjunctions, disjunctions and optional patterns; and(iv) It can be used with any modelling language.

In order to verify a workflow template, SPARQL verification queries are created based on the syntactic constraints. Two query forms are used in our work, including ASK and SELECT. According to [3], SELECT query is used to extract values, which are all, or a subset of the variables bound in a query pattern match, from a SPARQL endpoint. The variables that contain the return values are listed after a SELECT keyword. In the WHERE clause, "one or more graph patterns can be specified to describe the desired result" [6]. ASK query is used to return a boolean indicating whether a query pattern matches or not.

The following query[3] , for example, is used to check whether there exist errors related to improper uses of control nodes or not. This query is used to detect if there are any deadlocks caused by the combination of pairs of control nodes, Xor-split and And-join.

```
SELECT distinct ?xorsplit ?andjoin
WHERE {
?xorsplit  rdf:type h:Xor-split
?andjoin rdf:type h:And-join
?t1 h:hasGuardFunction/h:hasActivity _:b1
?t2 h:hasGuardFunction/h:hasActivity _:b2
?xorsplit h:connectsPlace/h:connectsTrans ?t1
?xorsplit h:connectsPlace/h:connectsTrans ?t2
?t1 h:connectsPlace/h:connectsTrans ?andjoin
?t2 h:connectsPlace/h:connectsTrans ?andjoin
FILTER(?t1!=?t2)}
```

As a result of the execution of each SPARQL query created based on the syntactic constraints, we obtain an XML file which results in nodes consisting of required information (e.g., the name) and causes shortcomings. For example,

[3] The prefix is assumed as:
$PREFIX h :< http://www.semanticweb.org/CPNWF\# >$

Figure 5 shows the result of the execution of the above query applied to check whether the workflow, depicted in Figure 4, contains deadlocks or not.

```
<?xml version="1.0"?>
<sparql xmlns='http://www.w3.org/2005/sparql-results#'>
  ...
  <result>
    <binding name='xorsplit'>
      <uri>http://WFTemplate/AirlineBooking#t2</uri>
    </binding>
    <binding name='andjoin'>
      <uri>http://WFTemplate/AirlineBooking#t5</uri>
    </binding>
  </result>
  ...
</sparql>
```

Fig. 5: Checking deadlocks caused of the two control nodes $Xor - split$ and $And - join$

The query presented above does not only demonstrate that we can use the SPARQL query language to check the syntactic correctness of workflow processes, but also the useful of terminology provided by the CPN Ontology, such as *Xor-split* and *hasGuardFunction*.

## 5 Related Work

Today, the problem of ensuring the correctness of process models have been paid attention in various researches. However, researchers mainly focused on checking the compliance of models concerning aspects of the syntax and formal semantics. To process modelling, there exist some formal criteria, such as "soundness", "completeness", "well-structureness". These criteria are used to examine anomalies, e.g., deadlock, livelock, missing synchronization and dangling references. There are some methods have been proposed to verify workflow models, such as Petri Nets-based [7], [8], logic-based [9], [10], graph reduction-based [11] methods. However, most of them check the conformance of a workflow process based on the principle that if the constraints on data and control flow are met during execution, the workflow is correct.

In fact, the ontology-based approach for modelling business process is not a new idea [6]. In order to support (semi-)automatic system collaboration, some works, such as [12], [13], made efforts to build business workflow ontologies.

Machine-readable definitions of concepts and interpretable format, therefore, are provided via these ontologies. However, they do not mention the issues relating to a taxonomy of constraints and also the verification of workflows at the syntactic level.

In our work, the Web Ontology Language is used to develop the CPN Ontology for representing business processes modelled with CPNs. Our ontological approach enables the formulation of constraints added to the CPN Ontology to ensure the soundness of workflow patterns. The constraints are then applied to concrete CBWTs using an RDF engine in order to automatically verify workflow processes at the syntactic level.

## 6 Conclusion

In this paper, we introduce an ontological approach to support designers in verifying workflow processes. We shortly present the CPN Ontology, a representation of CPNs and OWL DL, which is defined to take advantage of powerful reasoning systems. Then, we describe two groups of constraints that ensure the soundness of well-formed workflow processes. We concentrate on defining axioms corresponding to the syntactic constraints and introduce some axioms involving the use of control nodes.

To verify concrete CBWTs, which are represented in RDF format, we specify the syntactic errors and errors related to improper uses of control nodes as SPARQL queries. By relying on the CORESE semantic engine, we show that the SPARQL query language is usable to workflow verification.

We know that checking workflow templates at build-time is not enough to ensure the proper execution of a workflow template. The ability to check the correctness of workflow execution is also needed. In our future work, a run-time environment is going to develop for workflow verification.

## References

1. Lu, S., Bernstein, A.J., Lewis, P.M.: Automatic workflow verification and generation. Theor. Comput. Sci. **353**(1-3) (2006) 71–92
2. Nguyen, T.H.H., Le-Thanh, N.: An ontology-enabled approach for modelling business processes. In: Beyond Databases, Architectures, and Structures. Volume 424 of Communications in Computer and Information Science. Springer International Publishing (2014) 139–147
3. W3C: Sparql 1.1 query language. http://www.w3.org/TR/sparql11-query/ (March 2013) W3C Recommendation.
4. Kristensen, L.M., Christensen, S., Jensen, K.: The practitioner's guide to coloured petri nets. STTT **2**(2) (1998) 98–132
5. Corby, O., et al.: Corese/kgram. https://wimmics.inria.fr/corese
6. Nguyen, T.H.H., Le-Thanh, N.: Ensuring the Semantic Correctness of Workflow Processes: An Ontological Approach. In Grzegorz J. Nalepa and Joachim Baumeister, ed.: Proceedings of 10th Workshop on Knowledge Engineering and Software Engineering (KESE10) co-located with 21st European Conference on Artificial

Intelligence (ECAI 2014). Volume 1289. CEUR Workshop Proceedings, Prague, Czech Republic (August 2014)
7. van der Aalst, W.M.P.: The application of petri nets to workflow management. Journal of Circuits, Systems, and Computers **8**(1) (1998) 21–66
8. Verbeek, H., Basten, T., van der Aalst, W.: Diagnosing workflow processes using woflan. The computer journal **44** (1999) 246–279
9. Bi, H.H., Zhao, J.L.: Applying propositional logic to workflow verification. Information Technology and Management **5**(3-4) (2004) 293–318
10. Wainer, J.: Logic representation of processes in work activity coordination. In: Proceedings of the 2000 ACM Symposium on Applied Computing - Volume 1. SAC '00, New York, NY, USA, ACM (2000) 203–209
11. Sadiq, W., Maria, Orlowska, E.: Analyzing process models using graph reduction techniques. Information Systems **25** (2000) 117–134
12. Koschmider, A., Oberweis, A.: Ontology based business process description. In: EMOI-INTEROP, Springer (2005) 321–333
13. Sebastian, A., Tudorache, T., Noy, N.F., Musen, M.A.: Customizable workflow support for collaborative ontology development. In: 4th International Workshop on Semantic Web Enabled Software Engineering (SWESE) at ISWC 2008. (2008)